

200207452-1

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15

UNITED STATES PATENT APPLICATION  
FOR  
STATUS-MESSAGE MAPPING

Inventor:

Bernd GUTJAHR

## FIELD OF THE INVENTION

The present invention relates generally to status-message mapping, and for example, to a method, an IT-infrastructure-management server, a computer program product and a propagated signal representing a program for mapping status messages of monitored objects to service elements in an IT-infrastructure-management system.

## BACKGROUND OF THE INVENTION

Due to the increasing complexity of modern information-technological (IT) networks (computer networks and telecommunication networks), integrated network management systems (also called "management platforms") have become an important tool for the operation of IT networks. For example, Hewlett-Packard offers such a management platform under the name "hp OpenView" (see, for example, hp OpenView Quick Reference Guide, February 2003), and a structure of a typical management platform is, for example, shown in H.-G. Hegering et al.: Integrated Management of Networked Systems, 1998, pp. 313). Conventionally, one task of "network management" has been the monitoring of the status, performance or availability of network elements and the provision of monitoring results, for example, to a network operator. Typically, network monitoring includes an alarm functionality which alerts the operator when an incident occurs which requires attention, for example, a fault, an impending fault, or a decline of performance or availability of a network element. A network element's status message on which such an alarm is based may either be received from a managed network element in response to (e.g. periodical) requests from a management server, or it may be sent asynchronously by the managed network element to the management server. Typically (but not necessarily) a network management system not only provides a flow of information from managed network elements to the management server (and, e.g. to the network operator), but also enables managed network elements to be manipulated, e.g. by the network operator

1 or, automatically, by a network management process. "Manipulating" may,  
2 for example, comprise configuring network elements or changing their exist-  
3 ing configuration, starting and stopping processes in network elements, allo-  
4 cating memory to them, etc. Known management architectures are, for ex-  
5 ample, the OSI management architecture (which is mainly used in the Tele-  
6 communications Management Network) and the Internet management archi-  
7 tecture which uses SNMP (Simple Network Management Protocol) as a com-  
8 munication model (see, for example, Hegering, pp. 121-196).

9 In simple network management systems, only hardware devices (such as  
10 network interconnect devices and end devices) are managed. However, what  
11 an operator of a network or client of a network provider is typically most in-  
12 terested in is not the functioning of the individual hardware devices and net-  
13 work connections, but rather the functioning of systems (i.e. combinations of  
14 several hardware components and an operating system used to utilize the  
15 hardware), applications and services provided by the network and its devices.  
16 Therefore, in more elaborate management systems, systems, applications and  
17 services are also managed (wherein "managed" often just means  
18 "monitored"). An application is typically an assembly of sub-applications on  
19 which it depends, which, in turn, depend on hardware resources (mostly on a  
20 multiplicity of hardware resources) and network connections. A service is a  
21 customer-based or user-based functionality which, for example, forms a part  
22 of a business process. It is typically composed of several application, operat-  
23 ing system, hardware and communication components and normally has an  
24 interface to the service receiver. Hence a service depends on service-sub-  
25 functionalities, which, in turn, depend on hardware and communication re-  
26 sources (mostly on a multiplicity of such resources). Usually, an application  
27 (such as an enterprise resource planning (ERP) application, an e-mail server  
28 application, a web server application etc.) may be considered as a "service",  
29 but the term "service" may also refer to higher-level objects which may, for  
30 example, be business-related (such as objects representing the different busi-  
31 ness fields of an enterprise). All these objects (hardware devices, systems,  
32 sub-applications/sub-services, applications/services) may be managed objects

1 of a network management system; they are all denoted as "service elements"  
2 hereinafter.

3 Examples of network management systems in which service elements  
4 are managed (monitored) are described in US 2002/0138638 A1 and US  
5 2002/0143788 A1.

6 Typically, there are dependencies between service elements: a fault of a  
7 hardware device will affect a sub-application (or a sub-service) which relies  
8 on this hardware device. In turn, a fault of a sub-application (or sub-service)  
9 will affect an application (or a service) relying on it. Such dependencies can  
10 be thought of as (virtual) links between service elements. Typically, these  
11 links are directed links since, for example, a hard-disk failure will affect a da-  
12 tabase application relying on this hard disk, but a failure of the  
13 (superordinate) database service will generally not affect the functionality of  
14 the (subordinate) hard disk. Similarly, the failure of a database management  
15 system (DBMS) (which may be considered as a "sub-application" here) will  
16 affect an ERP application (which may be considered as an "application") re-  
17 lying on the DBMS, but the (subordinate) DBMS will generally not be affected  
18 by a failure of the (superordinate) ERP application. The service elements are  
19 thus in relationships which may be modeled by an element graph having links  
20 (or edges) between service elements. The nodes of the graph represent the  
21 service elements. The links represent the dependencies, and the direction of a  
22 link may represent the direction of dependency. Thereby, higher-level and  
23 lower-level service elements are defined (wherein, of course, there may be  
24 more than two levels, as indicated by the example given above). In simple  
25 cases, such a graph will be tree-like (also called a "hierarchy"), with the root  
26 of the tree at, or above, the highest-level service element or elements. An  
27 example of a tree-like service-element graph is shown in Fig. 4 of US  
28 2002/0138638 A1. Generally, the element graph may have a non-tree-like  
29 structure (it may, for example, be a "lattice"), if more than one higher-level  
30 service depend on a lower-level service, i.e. it may have cycles. An example  
31 of such a non-tree-like service element graph is shown in Fig. 5 of US  
32 2002/0138638 A1.

1        In network management systems, there are generally two different kinds  
2 of status (or monitoring) messages of monitored objects: (i) asynchronous  
3 messages which are typically sent by management agents assigned to the  
4 monitored object to a management server, such as an SNMP trap in Internet  
5 management (see Hegering, pp. 171-179) or a CMIP notification in OSI man-  
6 agement (see Hegering, pp. 134-139); such asynchronous messages are sent  
7 by the management agents without request, e.g. triggered by an event de-  
8 tected by the agent in the monitored object; (ii) synchronous status mes-  
9 sages, i.e. responses of management agents returned in response to status  
10 requests, e.g. from a management server. Again, the message may be an  
11 SNMP response (issued in response to an SNMP request) in Internet man-  
12 agement or a CMIP response (issued in response to a CMIP GET operation) in  
13 OSI management. Both asynchronous and synchronous messages are generi-  
14 cally denoted as "status messages" hereinafter.

15        In contrast to most of today's network interconnect devices many of the  
16 applications available on the market are not (or, at least, not completely) in-  
17 strumented for application management; in particular, they are often not en-  
18 abled to provide information about their status to a management system via  
19 an agent, as interconnect devices can. Furthermore, certain high-level serv-  
20 ices may be virtual or logical elements which are not made up of a single ap-  
21 plication or resource, but of a (virtual) assembly of several applications and/or  
22 resources. For example, such a high-level service element might be a "credit  
23 department" of a bank, which relies on several applications. Typically, such  
24 virtual high-level services are not objects instrumented for management, ei-  
25 ther; in particular, they will not be able to send status messages to a man-  
26 agement system. Therefore, the current status of such uninstrumented appli-  
27 cations or high-level services is often not directly determined, but is indirectly  
28 concluded from the current status of lower-level elements on which it de-  
29 pends. For example, a failure of a network interconnect device may have an  
30 impact on applications and services on a server, for example a DNS server  
31 which can no longer be reached due to the failure, which, in turn, may influ-  
32 ence the status of higher-level applications and services relying on it. In order

1 to perform such IT-infrastructure management based on indirect conclusion, a  
2 representation of the service-element graph is established in the IT manage-  
3 ment system, and the (virtual) status of at least some of the service elements  
4 is affected by status messages received from other service elements on  
5 which they depend.

6 In an IT-infrastructure-management system with such a virtual service-  
7 element graph, as described in US 2002/0138638 A1, status messages from  
8 monitored objects are directly mapped to the service elements representing  
9 the respective monitored objects. Virtual service elements may depend on the  
10 status of the respective monitored object and be affected by a status change  
11 of it. As a consequence of a receipt of such a status-change-indicating mes-  
12 sage at the service element representing such a monitored object, the status  
13 of the service element may be changed and the status change (not the status  
14 message itself) may be propagated upwardly to also influence the status of  
15 the higher-level service elements. For example, if a certain virtual service  
16 element depends on the availability of a certain hardware resource, directing  
17 a status message indicating a failure of this resource to the representation of  
18 this resource in the service-element graph and upward propagating the status  
19 change may cause the status of the superordinate service element to change  
20 from a "normal" state to a "critical" state which indicates that the virtual  
21 service element is not (fully) available.

22 In the prior art, two ways to map status messages to service elements  
23 are known: (i) a status message is mapped to a service element at the lowest  
24 hierarchical level, typically to a representation of the monitored object from  
25 which the status message originates. The status message may influence the  
26 status of the lowest-level service element to which it is mapped. This status  
27 change (not the status message itself) is then propagated upwardly in the  
28 service-element graph according to predefined rules. An example of such a  
29 status-upward propagation is illustrated at the left-hand side of Fig. 6 of US  
30 2002/0138638 A1; (ii) if a status message does not specifically refer to a  
31 monitored device, but to a higher-level service element, it may be immediately  
32 directed to the higher-level service element, and may directly influence its

1 status. This may happen if an application is "instrumented" for management.  
2 An example of this is illustrated at the right-hand side of Fig. 6 of  
3 US 2002/0138638 A1. In both cases, the service elements of the service-  
4 element graph are identifiable by a unique service-element identifier; and each  
5 status message carries with it a service-element identifier referencing the  
6 service element for which the message is destined.

## 7 8 SUMMARY OF THE INVENTION 9

10 A first aspect of the invention is directed to a method of mapping status  
11 messages of monitored objects to service elements in an IT-infrastructure-  
12 management system. In the method according to the first aspect, the service  
13 elements and their dependencies are represented by an element graph having  
14 directed links between service elements, thereby defining higher-level and  
15 lower-level service elements. The method comprises: directing a status mes-  
16 sage to at least one higher-level service element; ascertaining, at the higher-  
17 level service element, whether the status message pertains to a lower-level  
18 service element connected with the higher-level service element; downwardly  
19 propagating of the status message to said lower-level service element in re-  
20 sponse to a positive outcome in said ascertaining.

21 According to another aspect, a method is provided of mapping status  
22 messages of monitored objects to service elements in an IT-infrastructure-  
23 management system. The service elements and their dependencies are repre-  
24 sented by an element graph having directed links between service elements,  
25 thereby defining higher-level and lower-level service elements. The method  
26 comprises: analyzing a status message of a monitored object, and adding at-  
27 tributes to the status message related to information contained in the status  
28 message, directing the status message to at least one higher-level service  
29 element; ascertaining, at the higher-level service element, on the basis of at  
30 least one of the attributes, whether the status message pertains to a lower-  
31 level service element connected with the higher-level service element; down-  
32 wardly propagating of the status message to said lower-level service element

1 in response to a positive outcome in said ascertaining.

2 According to another aspect, an IT-infrastructure-management server is  
3 provided arranged to map status messages of monitored objects of the IT in-  
4 frastructure to service elements which are represented in the server in an  
5 element graph having directed links connecting service elements, thereby de-  
6 fining higher-level and lower-level service elements. The server is pro-  
7 grammed to: direct a status message to at least one higher-level service ele-  
8 ment; ascertain, at the higher-level service element, whether the status mes-  
9 sage pertains to a lower-level service element connected with the higher-level  
10 service element; propagate downwardly the status message to said lower-  
11 level service element in response to a positive outcome in said ascertaining.

12 According to another aspect, an IT-infrastructure-management server is  
13 provided arranged to map status messages of monitored objects of the IT in-  
14 frastructure to service elements which are represented in the server in an  
15 element graph having directed links connecting service elements, thereby de-  
16 fining higher-level and lower-level service elements. The server is pro-  
17 grammed to: analyze a status message of a monitored object, and add attrib-  
18 utes to the status message related to information contained in the status  
19 message, direct the status message to at least one higher-level service ele-  
20 ment; ascertain, at the higher-level service element, on the basis of at least  
21 one of the attributes, whether the status message pertains to a lower-level  
22 service element connected with the higher-level service element; propagate  
23 downwardly the status message to said lower-level service element in re-  
24 sponse to a positive outcome in said ascertaining.

25 According to another aspect, a computer program product is provided. It  
26 comprises a machine-readable medium with program code stored on it. The  
27 program code, when executed on a computer system, carries out a method  
28 of mapping status messages of monitored objects to service elements in an  
29 IT-infrastructure-management system. The service elements and their de-  
30 pendencies are represented by an element graph having directed links be-  
31 tween service elements, thereby defining higher-level and lower-level service  
32 elements. The program code is arranged to: direct a status message to at

1 least one higher-level service element; ascertain, at the higher-level service  
2 element, whether the status message pertains to a lower-level service ele-  
3 ment connected with the higher-level service element; downwardly propa-  
4 gating the status message to said lower-level service element in response to a  
5 positive outcome in said ascertaining.

6 According to another aspect, a computer program product is provided. It  
7 comprises a machine-readable medium with program code stored on it. The  
8 program code, when executed on a computer system, carries out a method  
9 of mapping status messages of monitored objects to service elements in an  
10 IT-infrastructure-management system. The service elements and their de-  
11 pendencies are represented by an element graph having directed links be-  
12 tween service elements, thereby defining higher-level and lower-level service  
13 elements. The program code is arranged to: analyze a status message of a  
14 monitored object, and add attributes to the status message related to infor-  
15 mation contained in the status message, direct the status message to at least  
16 one higher-level service element; ascertain, at the higher-level service ele-  
17 ment, on the basis of at least one of the attributes, whether the status mes-  
18 sage pertains to a lower-level service element connected with the higher-level  
19 service element; propagate downwardly the status message to said lower-  
20 level service element in response to a positive outcome in said ascertaining.

21 According to another aspect, a propagated signal carried on an electro-  
22 magnetic waveform is provided. The signal comprises a representation of  
23 program code. The program code, when executed on a computer system,  
24 carries out a method of mapping status messages of monitored objects to  
25 service elements in an IT-infrastructure-management system. The service  
26 elements and their dependencies are represented by an element graph having  
27 directed links between service elements, thereby defining higher-level and  
28 lower-level service elements. The program code is arranged to: direct a status  
29 message to at least one higher-level service element; ascertain, at the higher-  
30 level service element, whether the status message pertains to a lower-level  
31 service element connected with the higher-level service element; downwardly  
32 propagating the status message to said lower-level service element in re-

1 sponse to a positive outcome in said ascertaining.

2 According to another aspect, a propagated signal carried on an electro-  
3 magnetic waveform is provided. The signal comprises a representation of  
4 program code. The program code, when executed on a computer system,  
5 carries out a method of mapping status messages of monitored objects to  
6 service elements in an IT-infrastructure-management system. The service  
7 elements and their dependencies are represented by an element graph having  
8 directed links between service elements, thereby defining higher-level and  
9 lower-level service elements. The program code is arranged to: analyze a  
10 status message of a monitored object, and add attributes to the status mes-  
11 sage related to information contained in the status message, direct the status  
12 message to at least one higher-level service element; ascertain, at the higher-  
13 level service element, on the basis of at least one of the attributes, whether  
14 the status message pertains to a lower-level service element connected with  
15 the higher-level service element; propagate downwardly the status message  
16 to said lower-level service element in response to a positive outcome in said  
17 ascertaining.

18 Other features are inherent in the methods and products disclosed or  
19 will become apparent to those skilled in the art from the following detailed  
20 description of embodiments and its accompanying drawings.

## 21 DESCRIPTION OF THE DRAWINGS

22  
23  
24 Embodiments of the invention will now be described, by way of exam-  
25 ple, and with reference to the accompanying drawings, in which:

26 Fig. 1 illustrates an exemplary embodiment of a managed IT infra-  
27 structure including a management server;

28 Fig. 2 is an exemplary service model based on the IT infrastructure  
29 of Fig. 1;

30 Fig. 3 illustrates message preprocessing considering two exemplary  
31 messages;

32 Fig. 4 is a flow diagram of a message-mapping process;

Fig. 5 illustrates how messages may be handled when the service model is still undergoing an evolution;

Fig. 6 is a high-level architecture diagram of a part of a management platform arranged to map management messages to service elements;

Fig. 7 is a diagrammatic representation of a management server.

## DESCRIPTION OF THE PREFERRED EMBODIMENTS

Fig. 1 illustrates an exemplary managed IT network. Before proceeding further with the detailed description of Fig. 1, however, a few items of the embodiments will be discussed.

In the disclosed embodiments, not only physical devices (such as network interconnect devices and end devices), but also "logical" objects such as system applications and services are managed. As already mentioned at the outset, a system is typically a combination of several hardware components and an operating system used to utilize the hardware. An application is typically an assembly of sub-applications on which it depends, which, in turn, depend on hardware resources or systems. A service is typically a customer-based or user-based functionality which, for example, forms part of a business process. It is typically composed of several applications, operating system, hardware and communication components. Hence a service depends on service-sub-functionalities, which, in turn, depend on hardware and communication resources. Usually, an application (such as an ERP planning application, an e-mail server application, a web server application etc.) may be considered as a "service", but the term "service" may also refer to higher-level objects which may, for example, be business-related (such as objects representing the different business fields of an enterprise). All these physical objects (hardware devices, network connections) and logical objects (sub-applications/sub-services, applications/services) may be managed objects of an IT-infrastructure-management system; they are all denoted as "service elements" hereinafter.

Depending on the actual physical and logical topology of the IT infra-

1 structure managed, there are dependencies between service elements: a fault  
2 of a hardware device will affect a sub-application (or a sub-service) which  
3 relies on this hardware device. In turn, a fault of a sub-application (or sub-  
4 service) will affect an application (or a service) relying on it. Such dependen-  
5 cies, or precedence relationships, can be thought of as (virtual or logical)  
6 links between service elements. Typically, these links are directed links since,  
7 for example, a hard-disk failure will affect a DBMS relying on this hard disk,  
8 but a failure of the superordinate DBMS will generally not affect the function-  
9 ality of the subordinate hard disk. Similarly, the failure of a DBMS (which may  
10 be considered as a "sub-application" here) will affect an ERP application  
11 (which may be considered as an "application") relying on the DBMS, but the  
12 subordinate DBMS will generally not be affected by a failure of the superordi-  
13 nate ERP application. The service elements are thus related to one another  
14 according to a virtual service-element lattice which defines links connecting  
15 the service elements; the links are directed to represent the dependency of  
16 the connected service elements. Thereby, higher-level and lower-level service  
17 elements are defined (wherein, of course, there may be more than two levels,  
18 as indicated by the example given above). The service element lattice is also  
19 called "the service model".

20 Typically, many higher-level service elements are not enabled (or  
21 "instrumented") to provide the management system with information about  
22 their status. Consequently, the current status of such service elements can-  
23 not (or not completely) be directly determined, but only indirectly inferred  
24 from the current status of other (mostly lower-level) elements on which they  
25 depend. In order to include virtual service elements and non-instrumented  
26 service elements in the management process and to perform said status in-  
27 ference, the service-element lattice is represented in a memory of the IT  
28 management system.

29 The service model can be described by a directed graph. For example  
30 the nodes of the graph represent the service elements, and its edges repre-  
31 sent the links connecting the service elements, wherein their direction repre-  
32 sents the dependency relationship between them. In simple cases, such a

graph will be tree-like (i.e. a hierarchy), with the root of the graph at, or above, the highest-level service element or elements. Generally, the graph may have a non-tree-like structure (i.e. it may have cycles), if more than one higher-level service depend on one and the same lower-level service. However, the orientation of the directed links is such that the graph has no directed cycles since, typically, a service element, does not, directly or indirectly, affect itself. A representation of the graph which can be easily processed in the computer, for example, is the so-called adjacency-list representation or the adjacency-matrix representation (see, for example, R. Sedgewick: Algorithms in C, 1990, pp. 415 to 449).

As will be explained in more detail below, in some embodiments with only one highest-level element (which may only be the root of the graph, but need not represent a particular service) the status messages are initially directed to the highest-level element. In other embodiments with several highest-level service elements, all the status messages may initially be directed to all these highest-level service elements in parallel. The root element or the several highest-level elements form one or more "entry points" for status messages. The links in the graph and their directions are such that all managed service elements represented in the graph are reachable by downward propagation from the entry point, or, in the case of more than one entry point, from the combination of all entry points. "Downwardly propagating" means processing the graph represented in the computer; if the direction of the links is chosen as the direction opposite to the direction of dependency (which goes from bottom to top), the graph is processed along the links in the link direction. In principle, the direction of the links may also be chosen in the dependency direction; the graph is then processed in the direction opposite to the link direction.

In the embodiments, there may be two different kinds of status (or monitoring) messages of monitored objects: (i) asynchronous messages which are typically sent by management agents assigned to the monitored objects to a management server, such as an SNMP trap or a CMIP notification; such asynchronous messages are sent by the management agents with-

1 out request, e.g. triggered by an event detected by the agent in the moni-  
2 tored object; (ii) synchronous status messages, i.e. responses of management  
3 agents returned in response to status requests, e.g. from a management  
4 server. The message may be an SNMP response (issued in response to an  
5 SNMP request), a CMIP response (issued in response to a CMIP GET opera-  
6 tion). Both asynchronous and synchronous messages are generically denoted  
7 as "status messages" hereinafter.

8 As mentioned above, all the status messages are initially directed to the  
9 entry point or points of the service element graph. If the graph has been en-  
10 tered at such an entry point (which may depend on a "node condition") it is  
11 ascertained whether the respective status message pertains to a lower-level  
12 service element connected with the "entry node" considered, as will be ex-  
13 plained below. If the answer is positive, the status message is downwardly  
14 propagated to said lower-level service element.

15 Depending on the actual service-element structure, the service elements  
16 will generally be arranged in more than one level. The actions of ascertaining  
17 and downwardly propagating are then repeatedly carried out downwardly  
18 from level to level.

19 In some of the embodiments, conditions are associated (i) with links  
20 connecting the higher-level service elements with the next lower-level service  
21 elements; and/or (ii) with the service elements. The first-mentioned conditions  
22 are called "edge conditions" since the links they are associated with are  
23 edges of the service-element graph. The second-mentioned conditions are  
24 called "node conditions" since the service elements they are associated with  
25 are nodes of the graph. In the graph representation in the computer, the edge  
26 and node conditions are extra information associated with the edges and  
27 nodes. The edge and node conditions may, for example, be directly associ-  
28 ated with their links and edges, or indirectly by associating references with  
29 the links and edges pointing to the link and edge conditions. The edge condi-  
30 tions, or references to the edge conditions may, for example, be put in auxil-  
31 iary arrays indexed by edge number, or in adjacency-list nodes (if an adja-  
32 cency-list representation is used (see Sedgewick, p. 423)). The node condi-

1 tions, or references to the node conditions can, for example, be put in auxil-  
2 iary arrays indexed by node number or by making an array of "artificial  
3 nodes" an array of records which includes the node conditions or the refer-  
4 ences to them (if an adjacency-list representation is used (see Sedgewick, pp.  
5 423 and 421)).

6 In some of the embodiments, ascertaining whether the status message  
7 pertains to a lower-level service element connected to the higher-level service  
8 element includes testing the edge condition associated with the link to the  
9 lower-level service element. The message is only then propagated down-  
10 wardly along this edge if the edge condition is fulfilled. This means that, if  
11 there is no link with a fulfilled edge condition directed downwardly from a  
12 certain service element considered, the downward propagation of the status  
13 message is terminated at this service element. If there is no other service  
14 element in the entire service-element graph at which the status message is  
15 pending and from which it can further be downwardly propagated, the  
16 downward propagation of the message is globally terminated.

17 In other embodiments, the, or some of the, edge conditions are imple-  
18 mented as conditions associated with the nodes to which the respective links  
19 lead, i.e. as node conditions. The message is only then propagated down-  
20 wardly along an edge to the connected lower-level node if the node condition  
21 at the lower-level node is fulfilled.

22 As mentioned above, in some of the embodiments, there is not only one  
23 high-level service element which serves as an entry point for the status mes-  
24 sages, but several service elements that are typically at the same hierarchic  
25 level and form parallel entry points. In these embodiments, the status mes-  
26 sages are initially directed to all service elements forming such entry points.  
27 Since there is no link leading to these entry points from a higher level, condi-  
28 tions are associated with these entry point service elements. Hence, these  
29 conditions are node conditions. They are tested when a status message is  
30 directed to the associated entry point. The message only enters the graph at  
31 this entry point if its node condition is fulfilled. As a consequence, the action  
32 of testing edge conditions, and, if applicable, downwardly propagating the

1 status message only commences at those entry points whose node conditions  
2 are fulfilled.

3 In some embodiments, the downward-propagation decisions are partly or  
4 completely taken on the basis of the original status message, as they are re-  
5 ceived from the agents. In other embodiments, however, a preprocessing of  
6 the status messages is carried out before they are directed to the entry point  
7 or points of the service-element graph, and the downward-propagation deci-  
8 sions are, at least partly, taken on the basis of data added to the status mes-  
9 sage as a result of the preprocessing. The preprocessing includes analyzing  
10 the status message, and, based on this analysis, adding attributes to the  
11 status message that are related to information contained in the original status  
12 message. The attributes, for example, are indicative of the service element or  
13 elements the status message pertains to. The down-propagation decisions,  
14 i.e. the actions of ascertaining whether the status message fulfills node  
15 and/or edge conditions, are then based on these attributes. This facilitates  
16 the downward propagation of the status messages.

17 As mentioned in the background section, in the prior art a service-  
18 element identifier is typically assigned to each service element, and all status  
19 messages carry with them a service-element identifier indicating which serv-  
20 ice element the message is destined for. This normally means that the service  
21 model has already to be known when the agents of the monitored objects  
22 (which generate the status messages) are configured. Similarly, when the  
23 service model is modified, the configuration of the agents (in particular that  
24 part of the configuration which defines the service-element identifier to be  
25 put in the status messages) has to be adapted to the modified model. How-  
26 ever, there is a desire to instrument the status messages (i.e. to arrange the  
27 agents to generate status messages) already in an early phase when the  
28 service model has not yet been defined. Hence, the use of service-element  
29 identifiers requires a great deal of pre-planning of the service model, it is  
30 therefore an obstacle to an evolutionary development (also called "rapid pro-  
31 tototyping") of the service model. Furthermore, it is arduous to adapt and re-  
32 install or update the agents of the monitored objects each time the service

1 model has been modified.

2 By contrast, in the embodiments all the status messages are initially ei-  
3 ther directed to the top of the service-element graph, i.e. to its root, or to a  
4 few highest-level service elements just below the root. This eliminates the  
5 need to include in a status message a service-element identifier referencing  
6 the specific service-element the message is destined for. Accordingly, in the  
7 embodiments, the status messages either have no service-element identifier  
8 at all, or only a "high-level identifier" which indicates the high-level entry  
9 point or points to which all the status messages are directed. This means that  
10 either nothing, or only the entry point, has to be defined at the time when the  
11 agents are installed. The detailed service model can be defined, extended,  
12 modified and refined at a later stage in an evolutionary process. The fact that  
13 the status messages are not directed to the service elements for which they  
14 are destined by means of a service-element identifier obviates the need to  
15 permanently adapt the status-message-producing agents in such an evolu-  
16 tionary process. Rather, the mapping of the status messages to the service  
17 elements for which they are destined is achieved by the conditions associated  
18 with the service model, as explained above.

19 A further issue with regard to the mapping of status messages to service  
20 elements in the prior art is that there is usually only one service-element iden-  
21 tifier per service element. However, a status message often influences more  
22 than one service element. One known solution is to provide the status mes-  
23 sages concerned with more than one service-element identifier. However, this  
24 increases the problems mentioned above. Another known solution is to create  
25 an additional element in the service model which has only the task to propa-  
26 gate a status change represented by the status message to the several serv-  
27 ice elements influenced by it; the status message is then only directed to this  
28 additional element. However, this solution renders the service model compli-  
29 cated.

30 By contrast, in the embodiments the status messages may be mapped to  
31 several service elements without a need to carry service-element identifiers of  
32 these elements or to introduce additional elements in the service model. In

1 the embodiments, a mapping of status messages to more than one service  
2 element is achieved by defining the edge and node conditions associated with  
3 the service-element hierarchy such that the propagation path of the status  
4 messages is branched so that they are propagated to the several service ele-  
5 ments concerned.

6 The conventional status-message mapping by means of service-element  
7 identifiers has good scaling behavior; for example, doubling the service model  
8 (e.g. by adding a partial service-element graph comparable in size with the  
9 existing one) would not require additional performance, if the conventional  
10 identifier-based mapping is used. In another (conceivable) solution, however,  
11 in which message attributes or the message itself are tested at each service  
12 element, the required performance to map the messages would be doubled.  
13 In the embodiments, however, adding a new top-level element with the new  
14 partial graph below it would add only one additional test at the new top-level  
15 element (assuming that the message does not pertain to the new partial  
16 graph). Accordingly, the scaling behavior of the downward-propagation map-  
17 ping used in the embodiments is nearly as good as in conventional identifier-  
18 based mapping.

19 Some of the embodiments of the computer program product with pro-  
20 gram code for performing the described methods include any machine-  
21 readable medium that is capable of storing or encoding the program code.  
22 The term "machine-readable medium" shall accordingly be taken to include,  
23 for example, solid state memories and, removable and non removable, optical  
24 and magnetic storage media. In other embodiments, the (computer program)  
25 product is in the form of a propagated signal comprising a representation of  
26 the program code, which more and more becomes the usual way to distribute  
27 software. The signal is carried on an electromagnetic wave, e.g. a radio wave  
28 transmitted over a copper cable or through the air, or a light wave transmit-  
29 ted through an optical fiber. The program code may be machine code or an-  
30 other code which can be converted into machine code, such as source code  
31 in a multi-purpose programming language, e.g. C, C + +, Java, C#, etc. The  
32 embodiments of a server computer include commercially available general-

purpose computers programmed with the program code.

Fig. 1: Managed IT infrastructure:

Returning now to Fig. 1, it illustrates an exemplary managed IT infrastructure 1. The IT infrastructure 1 is a (sub-)network connected to the Internet 2 and includes several end systems (or hosts) connected with one another and the Internet 2 by means of transmission media (e.g. cables) and network interconnect devices, such as routers 3. The exemplary end devices are: (i) an ERP database server 4 ("ERP" means "Enterprise Resource Planning"), (ii) an ERP user-interface (UI) server 5, (iii) three ERP user interfaces 6.1, 6.2, 6.3, (iv) four exchange servers 7.1, 7.2, 7.3, 7.4, and (v) a management server 8 with a management user interface 9. Two disk drives 10.1, 10.2 are associated with the ERP database server 4. The end systems host different applications; the ERP database server 4 and the ERP UI server 5 host parts 11.1, 11.2 of an SAP® system, and the exchange servers 7.1 to 7.4 host an exchange application 12.1 to 12.4, for example Microsoft® Exchange®. All managed network elements (i.e. all network elements except the management server 8) are equipped with a management agent 13 which is a remotely running program able to collect and send management information about the associated network element and/or application, either asynchronously or on request, to the management server 8, and, in certain cases, to manipulate the associated network element or application upon a request from the management server 8.

The management server 8 hosts a management platform 14 which will be described in more detail in connection with Fig. 6. In the framework of the management platform 14, a service model 15 of the IT infrastructure 1 is defined. The service model 15 has a structure of a graph; it is thus represented by a graph data structure in primary and/or secondary memory of the management server 8. One of the management platform's tasks is to request messages and receive the requested messages from the agents 13, or to receive unrequested (i.e. asynchronous) messages from them, map these mes-

sages to one or more elements of the service model 15, and, finally, perform a management activity based on the mapped message, for example, change the status of the service element to which the message has been mapped, propagate the status change upwardly to service elements depending on the element to which the message has been mapped, display in the management UI 9 that a critical change of a service element has occurred, cause an alarm, enter status information into metric values indicative of compliance with service level agreements (SLAs), e.g. a value indicative of "Quality of Service", etc.

Fig. 2: Service model:

Fig. 2 is a representation of an exemplary service-element graph, or service model, 15 of the exemplary IT infrastructure 1 of Fig. 1. In the management server, this graph, for example, may be represented by an adjacency-list or adjacency-matrix representation, as explained above. The nodes of the graph 15 are service elements 16, and the graph's edges represent dependency links 17 between the service elements 16. The links 17 are directed; in the example of Fig. 2 the link direction (which is illustrated by arrows in Fig. 2) is directed opposite to the actual dependency direction. In Fig. 2, higher-level service elements 16 are drawn above the lower-level service elements 16 on which they depend; therefore, the links 17 are directed downwardly.

Some of the service elements 16 correspond to network elements of Fig. 1, but not all network elements are necessarily represented in the service model 15. For example, the routers 3 of Fig. 1 are not expressly represented. Of course, a failure of a non-represented network element, such as a router 13, may affect a represented service element 16. On the other hand, the service model 15 includes logical service elements which are not direct representations of network elements, but are logical entities based on them.

The service model 15 has two highest-level service elements, an ERP (SAP) service 16.1 and an E-mail-exchange service 16.2. The ERP service

16.1 has two children, Database server (DB server) 16.3 and User-interface server (UI server) 16.4. The E-mail exchange service 16.2 has three children, the regions US 16.5, ASIA 16.6 and EUROPE 16.7. The DB server 16.3 has two children, DB1 16.8 and DB2 16.9. The UI server 16.4 has three children, UI1 16.10, UI2 16.11 and UI3 16.12. The region US 16.5 has two children, Exchange server ES1 16.13 and Exchange server ES2 16.14. The regions ASIA 16.6 and EUROPE 16.7 have one child each, exchange server ES3 16.15 and ES4 16.16. DB1 16.8 has two children, TABLE INVOICE 16.17 and TABLE client 16.18, and DB2 16.9 has one child, TABLE Employee 16.19. Finally, the service elements' TABLE INVOICE 16.17 and TABLE Client 16.18 have one common child, DISK1 16.20, and TABLE Employee 16.19 has one child, DISK2 16.21. The service elements DB Server 16.3, UI server 16.4, the Exchange servers ES1 to ES4, 16.13-16.16, and DISK1 16.20 and Disk 2 16.21 partly correspond to the physical network elements 4, 5, 7.1 to 7.4, 10.1 and 10.2 of Fig. 1. For example, the servers have a logical component, a server process, hosted on the physical servers. The remaining service elements 16 of Fig. 2 are logical elements.

All service elements 16 can be reached along the directed links 17 starting from the highest-level elements 16.1, 16.2. Although the links from DB1 16.8 to DISK1 16.20 via 16.17 and 16.18 form a loop, they are directed in parallel from DB1 16.8 to DISK1 16.20, so that there is no directed cycle (in other words, a "round trip" from DB1 16.8 to DISK1 16.20 and back to DB1 16.8 is excluded).

In some of the embodiments, the service model 15 has only one entry point, which need not represent a service element, but may be a virtual node, the only purpose of which is to serve as a unique entry point to the service model 15. In Fig. 2, this alternative is illustrated by a root 18 linked to the highest-level service elements 16.1 and 16.2.

Link conditions 19 are associated with the links 17. Furthermore, in embodiments without a root 18, node conditions 20 are associated with the highest-level nodes 16.1, 16.2. In Fig. 2, the link and node conditions 19, 20 are illustrated by rectangular boxes. A condition 19, 20, for example, may

1 have the form "variable operator value". The operator may be an equality or  
2 inequality operator or, if an order relation is defined on the data to be tested,  
3 a ">" or a "<" relation, etc. Composite conditions are also feasible. If a link  
4 condition 19 is true for a certain message, the message is downwardly  
5 propagated via the link 17 with which the condition 19 is associated to a  
6 downward service element 16 connected by the link 17; if it is not true, the  
7 message is not propagated via this link 17. If a node condition 20 is true for a  
8 certain message, the message is lead to the service element 16 with which  
9 the node condition 20 is associated; if it is not true, the message is not lead  
10 to that service element 16 (which means that, if the service element is one of  
11 the highest level service elements 16.1 and 16.2, the message does not enter  
12 the service model 13 at this service element).

13 In embodiments with a root 18, the role of the node conditions 20 at the  
14 highest-level service elements 16.1, 16.2 is taken by corresponding link ele-  
15 ments associated with the links between the root 18 and the highest-level  
16 service elements 16.1 and 16.2.

17 In other embodiments, lower-level service elements may also have node  
18 conditions associated, which replace the link conditions.

19  
20 Figs. 2 and 3: Message mapping:

21  
22 The mapping of messages to service elements by downwardly propa-  
23 gating the messages in the service model 15 is now illustrated by means of  
24 two exemplary messages 21a, 21b shown at the lower part of Figs. 3a and  
25 3b. As mentioned above, such messages 21 are synchronously or asynchro-  
26 nously produced by the agents 13 (Fig. 1) of managed network elements. As  
27 will be explained later in connection with Fig. 3, the messages 21 need not  
28 be the originally produced versions. Rather, the representation of the informa-  
29 tion contained in the messages may be changed, and further information may  
30 be added, in a preprocessing procedure. For example, information contained  
31 in the form of plain text may be extracted and added to the original message  
32 in the form of an attribute value. Furthermore, information about the structure

of the service model may be added, depending on information contained in the original message. The resulting messages are called "preprocessed messages". The exemplary messages 21a and 21b used to illustrate the process of downward propagation are such preprocessed messages. The first preprocessed message 21a has four attributes 22, "APPL", "NODE", "TEXT", and "REGION"; the values 23 of these attributes are "EXCHANGE", "ES2", "EXCHANGE SERVER IS DOWN ", and "US". The second exemplary message 21b has six attributes 22, "APPL", "OBJECT", "TEXT", "SUBSYSTEM", "TABLE", and "DISKPROBLEM". The corresponding attribute values 23 are "SAP", "DB1", "TABLE "INVOICE" FAILED: DISK FULL", "DB", "INVOICE" and "YES".

In embodiments without a virtual root 18, initially the node conditions 20 of all entry points 16.1, 16.2 are tested for all messages. For example, the node conditions 20 of the entry points 16.1, 16.2 test whether the attribute APPL has the value SAP (for entry point 16.1) or EXCHANGE (for entry point 16.2). For the first exemplary message 21a, only the node condition 20 at the entry point 16.2 is fulfilled; hence, the service model 15 is only entered by the first message 21a at the EXCHANGE entry point 16.2, but not at the SAP entry point 16.1.

In other embodiments with a virtual root 18, all messages 21 are entered at the root 18, but are only downwardly propagated to the next-level service elements 16.1 and/or 16.2 if link conditions, corresponding to the node conditions 20 of embodiments without root 18, are fulfilled. The result is the same as in embodiments without root 18, i.e. the first exemplary message 21a is only propagated to service element 16.2, but not to service element 16.1.

Then, message 21a is tested against three link conditions 19 associated with links 17 connecting the highest-level service element 16.2 with three service elements 16.5, 16.6 and 16.7 at the next-lower level. These link conditions 17 test whether the attribute REGION of the message 21a has the value US, ASIA or EUROPE. Since only the link condition "REGION = US" is fulfilled, the message is only downwardly propagated to the US service ele-

ment 16.5, but to none of the other linked service elements 16.6, 16.7. Now, the message 21a is tested against link conditions 19 associated with links 17 connecting service element 16.5 with the next-lower service elements, ES1 16.13 and ES2 16.14. These test whether the attribute NODE has the value ES1 or ES2. Since only the condition "NODE = ES2" is fulfilled, the message is only downwardly propagated via the link 17 with which this condition 19 is associated, to the ES2 service element 16.14, but is not downwardly propagated via the other link 17 to the ES1 service element 16.13. Since there is no further link connecting the ES2 service element 16.14 to another service element, the process of downward propagation is terminated. The service element at which the downward propagation is terminated is the service element for which the message 21a is destined. Hence, as a result, the message 21a is mapped to the service element for which it is destined. In Fig. 2 the path along which message 21a is propagated is illustrated by dual-line arrows.

Technically, the process of downwardly propagating a message means processing the service element graph with the message by starting from the highest-level entry points or the virtual root, traveling along the links in the link direction and thereby visiting all the nodes of the graph, provided that the link condition associated with a link under consideration is fulfilled, and terminating this process if there is no further link or fulfilled link condition.

In the embodiments shown, no unique service-element identifier, or service-element "key", that uniquely identifies the service element for which a message is destined (the "target" service element) is used in the message-mapping process. Hence, the messages 21 do not contain a service-element identifier. Rather, in order to enable a message 21 to be uniquely mapped, it is sufficient that the whole set of attributes 22 of a message 21, or a subset of them, uniquely identifies the target-service element in a composite manner. In the language used in connection with data modeling, entities which have no key attribute, such as the present service elements, are called "weak entities" and such a set of attributes that can uniquely identify a weak entity is also called a "partial key" (see, for example, R. Elmasri: Fundamentals of Da-

1 tabase Systems, 3<sup>rd</sup> edition, 2000, pp. 50, 59 and 60).

2 In certain circumstances, it may, however, be desired to propagate a  
3 message 21 to more than one lower-level service element 15. For example, in  
4 Fig.2, it might be desired to map message 21a to both ES1 16.13 and ES2  
5 16.14. Such a "branching" of a message 21 is achieved by defining the link  
6 conditions 19 in a way which does not limit the downward propagation to  
7 only one downward link 17. For example, if the link conditions 19 associated  
8 with the two links 17 connecting US 16.5 with ES1 16.13 and ES2 16.14 are  
9 replaced by the condition "REGION = US" (equal for both links), the message  
10 21a is propagated to both ES1 16.13 and ES2 16.14. In other embodiments,  
11 branching of a message 21 is achieved by using multi-valued attributes in the  
12 messages.

13 Downward propagation of the message 21 is terminated at those service  
14 elements 16 which either have no downward link 17 or have no downward  
15 link 17 with a fulfilled link condition 19. Hence, downward propagation of the  
16 first exemplary message 21a is terminated at service element ES2 16.14, or,  
17 in the example with the message branching, at both service elements ES1  
18 16.13 and ES2 16.14.

19 The purpose of the downward-propagation procedure is to map a mes-  
20 sage 21 to one or more service elements 16. In most of the embodiments,  
21 the mapping is defined such that a message 21 is mapped only to this service  
22 element 16 (or to those service elements 16) at which the downward propa-  
23 gation is terminated. In the example mentioned, the first exemplary message  
24 21a is then only mapped to service element ES2 16.14 (or, in the example  
25 with branching, it is only mapped to service elements ES1 16.13 and ES2  
26 16.14). Of course, not only the end node of the downward propagation to  
27 which the message is mapped may be affected by a status change indicated  
28 by the message, but the status change may also have an impact to the status  
29 of further service elements above them in the service-model hierarchy. Never-  
30 theless, in most of the embodiments, the messages are not additionally  
31 mapped to those affected higher-level service elements. Rather, the status of  
32 the lower-level service element to which the message has been mapped (but

1 not the message itself) is then upwardly propagated from it to the affected  
2 higher-level service elements, for example, as described in US 2002/0138638  
3 A1 mentioned at the outset.

4 In alternative embodiments, however, a message is not only mapped to  
5 the end nodes of the downward propagation, but also to some or all nodes  
6 lying above it on the propagation path. For example, a message 21a may also  
7 be mapped to the intermediate node US 16.5, as illustrated by a dashed dual-  
8 line arrow starting and ending at 16.5. The decision as to whether a message  
9 is also to be mapped to an intermediate node, or whether it is only to be  
10 handed over to the next lower-level nodes for which the link conditions are  
11 fulfilled, may be based on additional node conditions associated with the in-  
12 termediate nodes.

13 Downward propagation of the second exemplary message 21b is also  
14 illustrated by dual-line arrows in Fig. 2. As can be seen, message 21b only  
15 fulfills the node condition "APPL = SAP" associated with service element  
16 16.1, and therefore enters the service model 15 only at this service element  
17 16.1. Then, it is downwardly propagated to service elements 16.3, 16.8,  
18 16.17 and 16.20, as it only fulfills the corresponding link conditions  
19 "SUBSYSTEM = DB", "OBJECT = DB1", "TABLE = INVOICE" and  
20 "DISKPROBLEM = YES".

21 As mentioned above, the graph representing the service model 15 forms  
22 a loop including the service elements 16.8, 16.17, 16.18 and 16.20. Al-  
23 though the fact that the links from 16.8 to 16.20 are directed in a parallel  
24 manner prevents a message from being indefinitely propagated in a cycle,  
25 there are two different paths from 16.8 to 16.20, the first one via service  
26 element 16.17 and the second one via service element 16.18. For the sake of  
27 illustration let us assume now a particular embodiment in which the message  
28 is branched at 16.8 and downwardly propagated to both service elements  
29 16.17 and 16.18 (for example, by using link conditions at the links from 16.8  
30 to 16.17 and 16.18 which branch the message 21b, as explained above for  
31 the first exemplary message 21a). The two messages would then be further  
32 propagated to one and the same service element, 16.19. In order to prevent

1 such a "doubling" of messages, in some embodiments, a bookkeeping of the  
2 messages received at each node is performed, and if it is observed that a  
3 message has been "doubled", only one of the "doubled" messages is retained  
4 and further used. For example, if the messages are referenced by a message  
5 identifier, all nodes memorize the message identifiers of the messages already  
6 received, and discard an already memorized reference to a message. In Fig.  
7 2, the branching of message 21b at service element 16.8 and the downward  
8 propagation of the two messages to service element 16.19 is illustrated by  
9 additional dashed dual-line arrows. The discarding of the "doubled message"  
10 takes place at service element 16.19.

11 In the embodiment of Fig. 2, although the mapping of a message 21 in-  
12 volves testing composite conditions, the individual conditions 19, 20 tested  
13 at each stage of the downward propagation may be single conditions. This is  
14 due to the fact that the composite condition to be tested in the mapping pro-  
15 cedure is not processed at once, but rather in several stages when the serv-  
16 ice-element graph 15 is traversed. Of course, such single conditions are not  
17 mandatory; rather, composite conditions may also be used as link or node  
18 conditions, for example, conditions constructed by "AND", "OR" and/or  
19 "NOT" from single conditions.

20 Once a message 21 has been mapped to the target service element or  
21 elements 16, the actual management activity based on the mapped message  
22 will commence. For example, if the message indicates that a failure of the  
23 target-service element has occurred, the status of the target-service element  
24 is changed, its status change is upwardly propagated to service elements de-  
25 pending on the target-service element, the status changes are displayed in  
26 the management UI 9, an alarm is caused, the status change or failure infor-  
27 mation is entered into metric values indicative of compliance with SLAs, etc.

28  
29 Fig. 3: Message preprocessing:  
30

31 Fig. 3 illustrates message preprocessing considering the two exemplary  
32 messages already mentioned above. Although the original messages 24a, 24b

1 produced by the agents 13 (Fig. 1), in principle, contain all the information  
2 needed to uniquely map the messages 24a, 24b to service elements 16, the  
3 information is typically not represented by them in a form optimal for testing  
4 conditions as described in connection with Fig. 2. Two examples of original  
5 messages are shown in Fig. 2, 24a and 24b. For example, the original mes-  
6 sages 24a, 24b may not contain references to certain logical service elements  
7 16. For instance, original message 24a does not expressly include information  
8 identifying the REGION to which the message pertains. Furthermore, certain  
9 information to be tested may only be represented by them in an unstructured  
10 manner, for example, in the form of text. For instance, original message 24b  
11 indicates that the TABLE to which the message pertains is INVOICE, but this  
12 information is represented in the form of text, together with other text  
13 ("TABLE "INVOICE" FAILED: DISK FULL").

14 In some embodiments a message preprocessing (symbolized by box 25  
15 in Fig. 3) is carried out before the downward-propagation process. In the pre-  
16 processing 25 attributes 22 are added to the original messages 24, so that all  
17 the information to be tested in order to map the message to the target-service  
18 element is represented in the message in the form of attribute values. For ex-  
19 ample, during preprocessing of the first exemplary original message 24a, the  
20 value ES2 of the attribute NODE is read from original message 24a. An at-  
21 tribute-defining rule 26a defines which REGION attribute value corresponds to  
22 node ES2. According to the service model 15 (Fig. 2), the region attribute US  
23 depends on nodes ES1 and ES2. Hence, the attribute-defining rule 26a tests  
24 whether the NODE attribute is ES1 or ES2, and if the outcome is positive, the  
25 new REGION attribute is set to the value US. Hence, service-model informa-  
26 tion and information contained in the original message 24a is combined to  
27 create a new attribute; here the new REGION attribute with the value US is  
28 appended to the original message 24a to form the preprocessed message  
29 21a.

30 Similarly, during the preprocessing of the second exemplary original  
31 message 24b, the text contained in it is parsed. If, according to attribute-  
32 defining rules 26b the sub-strings "INVOICE" and "DISK FULL" are found,

1 new attributes TABLE and DISKPROBLEM are set to the values INVOICE and  
2 YES, and are appended to original message 24b to form the preprocessed  
3 message 21b. Of course, the attribute-defining rules 26 reproduced in Figs.  
4 3a and 3b are only fragments of the complete rules, rules which will, for ex-  
5 ample, test for the occurrence of other values in the NODE attribute, or the  
6 occurrence of other sub-strings in the parsed text, than the ones shown in  
7 Figs. 3a and 3b.

8 In further embodiments, both the "attribute creation", according to Fig.  
9 3a and the "attribute extraction" according to Fig. 3b are combined. In other  
10 words, some of the additional attributes are obtained by combining service-  
11 model information and message information, and others are extracted from  
12 unstructured information in the messages, such as text.

13 In some of the embodiments, original messages (rather than preproc-  
14 essed messages 21) are processed through the service element graph 15, in  
15 analogy to what has been explained in connection with Fig. 2. However,  
16 testing conditions are then more complicated, since, in the example of first  
17 original message 24a, the missing attribute REGION has to be inferred from  
18 the NODE attribute each time REGION is tested, and, in the example of sec-  
19 ond original message 24b, text has to be parsed each time a condition based  
20 on text information is evaluated.

21  
22 Fig. 4: Message preprocessing and mapping process:  
23

24 Fig. 4 is a flow diagram of the message preprocessing and mapping  
25 process described so far. A management message from a management agent  
26 is received at 27. At 28, the message is preprocessed, e.g. it is analyzed, and  
27 attributes are added to it, as described in connection with Fig. 3. In embodi-  
28 ments without a root 18, it is then tested at 29 for all nodes acting as an en-  
29 try point (called "entry nodes") whether the node condition associated with  
30 them is fulfilled. If the node condition is not fulfilled at a certain entry node,  
31 the service-element graph is not entered at this entry node, as indicated by  
32 box 30. If, however, the node condition of an entry node is fulfilled, the

1 service-element graph is entered at this entry node, at 31. It is then tested at  
2 32 for all downward links of the node under consideration whether the link  
3 condition associated with the downward link is fulfilled. If the link condition  
4 of a certain link is not fulfilled, the message is not propagated via this link, as  
5 indicated by box 33. However, if the link condition is fulfilled, the message is  
6 downwardly propagated via this link to the lower-level node connected by  
7 this link at 34. At 35, it is ascertained, at the lower-level node, whether the  
8 message has already been received, i.e. is a "duplicate message", and if so, it  
9 is removed. This description of the downward processing is recursive; i.e.,  
10 the activities 32 to 35 are repeatedly carried out to further propagate the  
11 message downwardly. The downward propagation stops at 32 if there is no  
12 downward link to be processed any more, or at 33, if there is no fulfilled link  
13 condition any more. In embodiments with a virtual root 18 acting as the entry  
14 node, the activities 29 and 30 may be omitted, the service-element graph is  
15 then entered at the root, at 31.

16  
17 Fig. 5: Evolution of service model:  
18

19 Fig. 5 illustrates how messages 21 may be handled when the service  
20 model 15 is still undergoing an evolution. At the left-hand side of Fig. 5, an  
21 exemplary case is shown in which the service model has not yet been defined  
22 for a certain application, here the EXCHANGE application 16.2, as may be the  
23 case in the beginning of a development phase. In the example shown, the  
24 not-yet-defined service model 15' only consists of the highest-level service  
25 element 16.2.

26 During an evolutionary process, the not-yet-defined service model 15'  
27 may be defined, or an already defined service model may be modified. At the  
28 right-hand side of Fig. 5, an exemplary result of such a service-model evolu-  
29 tion is shown, which corresponds to the EXCHANGE branch of Fig. 2 (and  
30 the above-mentioned embodiment in which message 21a (Fig. 3a) is not only  
31 mapped to the end node 16.14 of the propagation path, but also to the in-  
32 termediate node 16.5).

Although, at the beginning of the evolution, the service model 15' is not yet defined below the highest-level service element 16.2, the agents 13 of devices belonging to the not yet defined EXCHANGE service may already be instrumented to produce messages referring to the EXCHANGE service. The messages are then directed to the highest-level service element 16.2 (the message may be in the original or a preprocessed form, they are therefore denoted by "21/24" in Fig. 5). Already at this early stage of the service-model evolution, the instrumentation of the agents 13 may hence be made in a final form. Due to the fact that the messages are later downwardly propagated and that no service-element identifier needs to be contained in the messages to perform the mapping, no changes have later to be made to the structure and/or format in which the original messages are produced, and hence no changes have later to be made to the instrumentation of the agents 13, to adapt them to evolutionary extensions and modifications of the service model.

Of course, in the rudimentary state of the service model 15' illustrated at the left-hand side of Fig. 5, the messages directed to the entry node 16.2 are not downwardly propagated, since the service elements below the entry node 16.2 are not yet defined. Rather, all messages pertaining to the EXCHANGE service (i.e. messages whose APPL attribute has the value EXCHANGE) are only mapped to the entry node 16.2.

Only later, when the service model has evolved, for example, to the service model 15 at the right-hand side of Fig. 5, the messages are downwardly propagated in the service model 15 and mapped to lower-level service elements, for example to the service elements 16.5 and 16.14 as described above. Preprocessing rules 26 (Fig. 3) will also evolve with the evolution of the service model. However, the instrumentation of the agents 13, i.e. the rules according to which the agents 13 produce messages, need not be changed during the evolution. By contrast, in a mapping based on service-element identifiers, the agent instrumentation would have to be adapted to such an evolution, for example, since the (logical) service element 16.5 is not defined in the early stage of the evolution and hence would not yet have a

1 service-element identifier; only after service element 16.5 has been defined,  
2 the identifier referring to it could be included in the message-generation rules,  
3 which would require a reconfiguration of the agents.

4  
5 Fig. 6: Management platform:  
6

7 Fig. 6 is a high-level architecture diagram of the part of the management  
8 platform 14 (Fig. 1) concerned with the processing of the messages. The  
9 management platform 14, which is hosted by the management server 8 (Fig.  
10 1) includes a message preprocessor 36, a message mapper 37 and an event  
11 manager 38. The management platform 14 receives management messages  
12 from the agents 13 (Fig. 1) in the managed IT infrastructure 1. The message  
13 preprocessor 36 performs message preprocessing 25 (Fig. 3) according to  
14 preprocessing rules 26, as explained in connection with Fig. 3. The message  
15 mapper 37 performs the downward propagation of the preprocessed mes-  
16 sages in the service model 15, as described in connection with Fig. 2; it in-  
17 cludes a downward propagator 39 and stores a representation of the service  
18 model 15. As a result, the messages are mapped to one or more service ele-  
19 ments. The event manager 38 further processes the events represented by  
20 the messages based on the outcome of the event-mapping procedure; for ex-  
21 ample, it changes the status of a service element to which a message has  
22 been mapped, propagates the status upwardly to service elements depending  
23 on this element, displays that status changes of service elements have oc-  
24 curred in the management UI 9, causes an alarm, enters status information  
25 into values indicative of compliance with SLA's, etc.

26 In some embodiments, the message preprocessor 36, message mapper  
27 37, event manager 38, downward propagator 39 are software components  
28 separated from each other and from other software components, in other  
29 embodiments they are combined with each other and other software items.  
30 The terms "message preprocessor", "message mapper", "event manager"  
31 and "downward propagator" are hence functional terms, and their box-like  
32 representations in Fig. 6 are functional representations, which do not neces-

sarily imply that separate software components, solely devoted to message preprocessing, message mapping, event managing, and downward propagation are provided.

Fig. 7: Management server:

Fig. 7 is a diagrammatic representation of the management server 8 (Fig. 1) within which a set of instructions, for causing the management server to perform any of the methodologies discussed herein, may be executed. The management server 8 includes a processor 40, a main memory 41 and a network interface device 42, which communicate with each other via a bus 43. Optionally, it may further include a static memory 44 and a disk drive unit 45. A video display 46, an alpha-numeric input device 47 and a cursor control device 48 may form the management user interface 9 (Fig. 1). The network interface device 42 connects the management server 8 to the IT infrastructure 1 (actually, the management server 8 also is a part of the IT infrastructure 1). A set of instructions (i.e. software) 49 embodying any one, or all, of the methodologies described above, resides completely, or at least partially in or on a machine-readable medium, e.g. the main memory 41 and/or the processor 40. A machine-readable medium on which the software 49 resides may also be a data carrier 50 (e.g. a non-removable magnetic hard disk or an optical or magnetic removable disk) which is part of disk drive unit 45. The software 49 may further be transmitted or received as a propagated signal 51 via the Internet 2 and the IT network 1 through the network interface device 42.

Thus, a general purpose of the disclosed embodiments is to provide improved methods and products for mapping status messages elements of an IT-infrastructure-service model.

All publications and existing systems mentioned in this specification are herein incorporated by reference.

Although certain methods and products constructed in accordance with

1 the teachings of the invention have been described herein, the scope of cov-  
2 erage of this patent is not limited thereto. On the contrary, this patent covers  
3 all embodiments of the teachings of the invention fairly falling within the  
4 scope of the appended claims either literally or under the doctrine of equiva-  
5 lents.